Laboratory

2

Implementation of Cyclic Redundancy Check

Objective

The Data Link Layer is the second layer in the OSI model, above the Physical Layer, which ensures that the error free data is transferred between the adjacent nodes in the network. It breaks the datagram passed down by above layers and converts them into frames ready for transfer. This is called Framing. It provides two main functionalities

- Reliable data transfer service between two peer network layers
- Flow Control mechanism, which regulates the flow of frames such that data congestion is not there at slow receivers due to fast senders.

There are two basic strategies for dealing with errors. One way is to include enough redundant information (extra bits are introduced into the data stream at the transmitter on a regular and logical basis) along with each block of data sent to enable the receiver to deduce what the transmitted character must have been. The other way is to include only enough redundancy to allow the receiver to deduce that error has occurred, but not which error has occurred and the receiver asks for a retransmission. The former strategy uses Error-Correcting Codes and latter uses Error-detecting Codes.

Overview

This Cyclic Redundancy Check is the most powerful and easy to implement technique. Unlike checksum scheme, which is based on addition, CRC is based on binary division. In CRC, a sequence of redundant bits, called cyclic redundancy check bits, are appended to the end of data unit so that the resulting data unit becomes exactly divisible by a second, predetermined binary number. At the destination, the incoming data unit is divided by the same number. If at this step there is no remainder, the data unit is assumed to be correct and is therefore accepted. A remainder indicates that the data unit has been damaged in transit and therefore must be rejected.

1. Bit strings are created as representation of polynomials with coefficients ‘0’ and ‘1’only.

2. A k-bit frame is regarded as coefficients list for a polynomial with ‘k’ terms (x_{k-1} to x_0)

Eg: $x_5 + x_4 + x_0 = 110001$

When this method is used, the sender and the receiver should agree upon a generator polynomial, G(x) in advance.
Both the high and low order bits of G(x) must be '1'
To compute checksum for some frame with 'm' bits (polynomial = M(x), append 'r' zero bits to the lower end of the frame (r = degree of the generator polynomial) so that this check summed frame is divisible by G(x).
Divide M(x) by G(x) using modulo-2 division and subtract the remainder from M(x) using modulo-2subtraction. let the resultant be called as T(x)
T(x) is passed to the receiver and the receiver divides it by G(x).
If there is a remainder, there has been a transmission error.

Eg: frame = 1101011011
G(x) = $x_4 + x + 1$ = 10011
è degree = 4
Therefore, frame = 1101011011 + 0000
èM(x) = 11010110110000

Commonly used divisor polynomials are:
CRC 12 : $x_{12} + x_{11} + x_3 + x_2 + x + 1$
CRC 16 : $x_{16} + x_{15} + x_2 + 1$
CRC CCITT : $x_{16} + x_{12} + x_5 + 1$

Procedure

Implement the error correcting code Cyclic Redundancy Check (CRC) of data link layer using various polynomials like CRC-CRC 12, CRC 16 and CRC CCIPP.

Include header files stdio.h, string.h

```
#define N strlen(g)
char t[28],cs[28],g[28];
int a,e,c,b;
void xor()
{
for(c=1;c<N;c++)
cs[c]=((cs[c]==g[c])?'0':'1');
}
void crc()
{
for(e=0;e<N;e++)
cs[e]=t[e];
do
{
if(cs[0]=='1')
xor();
```

```
for(c=0;c<N-1;c++)
cs[c]=cs[c+1];
cs[c]=t[e++];
}while(e<=a+N-1);
}
int main()
{
int flag=0;
do{
printf("\n1.crc12\n2.crc16\ncrc ccit\n4.exit\n\nEnter your option.");
scanf("%d",&b);
switch(b)
{
case 1:strcpy(g,"1100000001111");
break;
case 2:strcpy(g,"11000000000000101");
break;
case 3:strcpy(g,"10001000000100001");
break;
case 4:return 0;
}
printf("\n enter data:");
scanf("%s",t);
printf("\n-----------------------\n");
printf("\n generating polynomial:%s",g);
a=strlen(t);
for(e=a;e<a+N-1;e++)
t[e]='0';
printf("\n--------------------------\n");
printf("mod-ified data is:%s",t);
printf("\n-----------------------\n");
crc();
printf("checksum is:%s",cs);
for(e=a;e<a+N-1;e++)
t[e]=cs[e-a];
printf("\n-----------------------\n");
printf("\n final codeword is : %s",t);
printf("\n------------------------\n");
printf("\ntest error detection 0(yes) 1(no)?:");
scanf("%d",&e);
if(e==0)
{
do{
printf("\n\tenter the position where error is to be inserted:");
scanf("%d",&e);
}
while(e==0||e>a+N-1);
```

```
t[e-1]=(t[e-1]=='0')?'1':'0';
printf("\n-----------------------\n");
printf("\n\terroneous data:%s\n",t);
}
crc();
for(e=0;(e<N-1)&&(cs[e]!='1');e++);
if(e<N-1)
printf("error detected\n\n");
else
printf("\n no error detected \n\n");
printf("\n-----------------------");
}while(flag!=1);
}
crc();
for(e=0;(e<N-1)&&(cs[e]!='1');e++);
if(e<N-1)
printf("error detected\n\n");
else
printf("\n no error detected \n\n");
printf("\n-----------------------");
}while(flag!=1);
```

Out Put

```
1.crc12
2.crc16
3.crc ccit
4.exit
Enter your option.1
enter data:1100110011100011
-----------------------
generating polynomial:1100000001111
-------------------------
mod-ified data is:11001100111000110000000000001100000001111
-----------------------
checksum is:1101110110001
-----------------------
final codeword is : 11001100111000111101110110001100000001111
------------------------
test error detection 0(yes) 1(no)?:1
no error detected
-----------------------
1.crc12
2.crc16
3.crc ccit
4.exit
```

Enter your option.2
enter data:11001100111000
---
generating polynomial:11000000000000101
---
mod-ified data is:110011001110000000000000000000000000000000000101

Viva Questions:

1. What is error control?
2. What is CRC?
3. What is Check Sum?
4. What are the Steps involved in CRC?
5. Explain the CRC with an example?
6. What are the values of M(X), G(x) values in this program?
7. What are the functionalities of Data Link Layer?

Lab Report

After successful completion of this lab, the student will have the ability to Know error control and detection method.